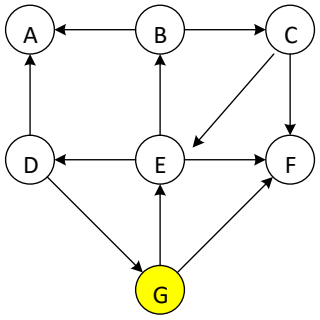
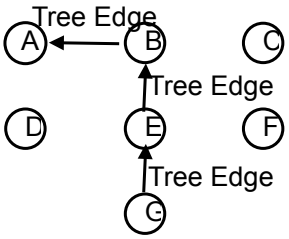
Assignment 6

Problem 1 (10 points). Run DFS on the following graph beginning at node G and show the sequence of nodes generated by the search. When you have two or more choices as the next node to visit, choose them in the alphabetical order.



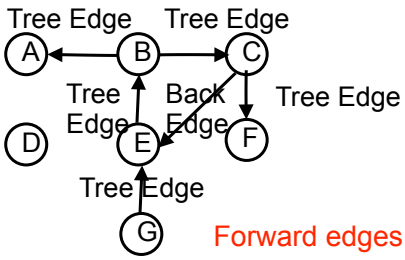
After completing the DFS, classify each edge as a *tree edge,* a *forward edge,* a *back edge,* or a *cross edge*.

G => E => B => A
Backtrack to B



A directed graph
Start at vertex G

B => C => F
Backtrack to C => B => E

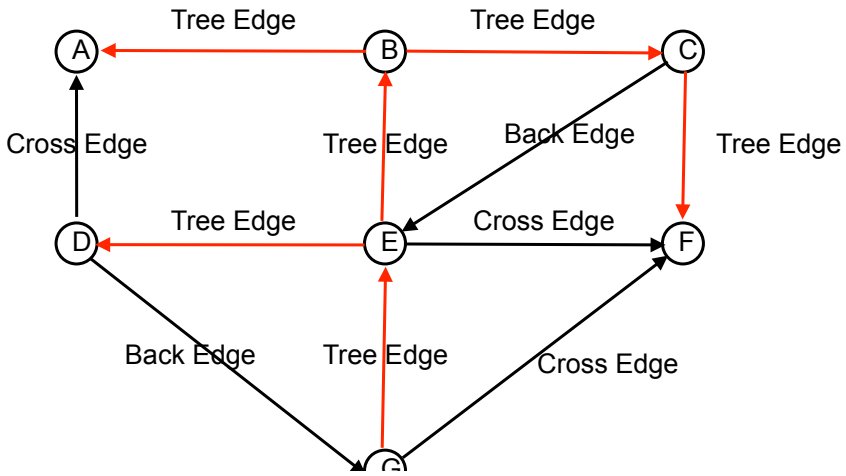


Forward edges: {(E, F), (G, F)}

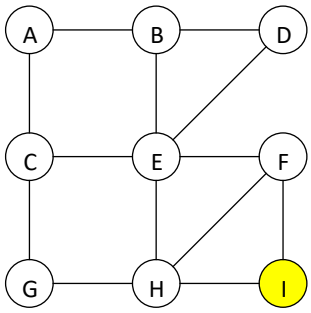
Back edges: connects a vertex to its ancestor
Forward edges: connects a vertex to its descendant
Cross edge: connects a vertex to a vertex that is neither its ancestor nor descendant

E => D
Finished

Result:
G => E => B => A => C => F => D

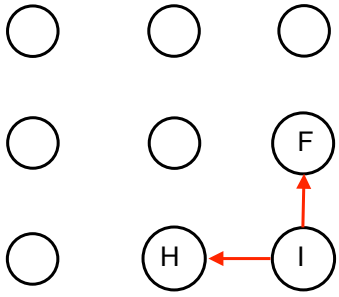


Problem 2 (10 points). Run BFS on the following graph beginning at node I and show the sequence of nodes generated by the search. When you have two or more choices as the next node to visit, choose them in the alphabetical order.

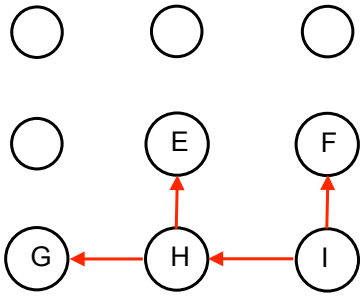


Start at vertex I

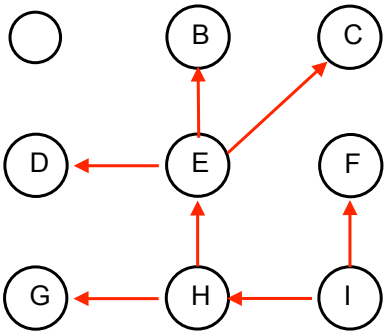
Explore vertices that are one-edge away from I



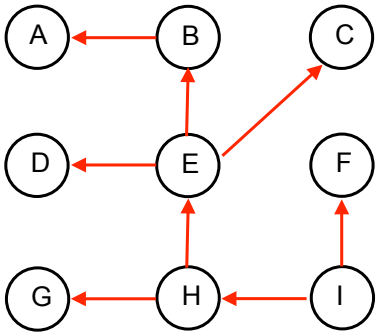
Explore vertices that are two-edge away from I



Explore vertices that are three-edge away from I

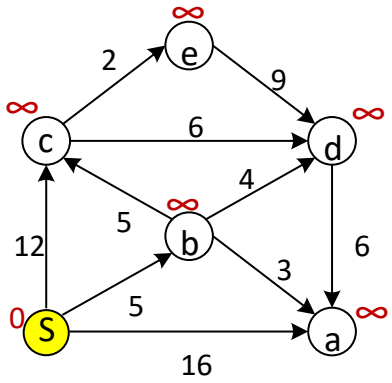


Explore vertices that are four-edge away from I



Result:
{I, F, H, E, G, B, C, D, A}

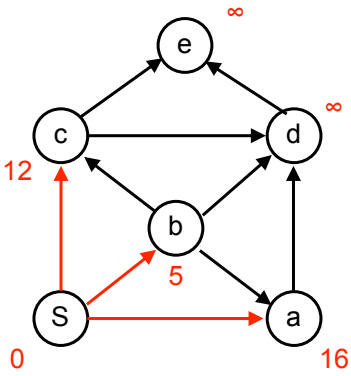
Problem 3 (10 points). Run Dijkstra's algorithm on the following graph beginning at node S.



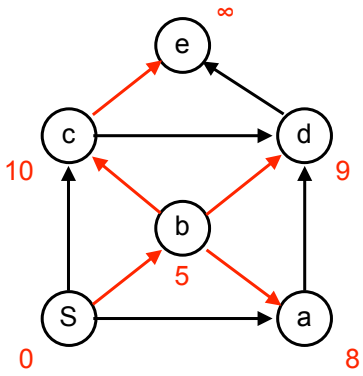
Initially, are in Q, C is empty,
all vertices D[s] = 0,
and D[v] = ∞ for all other vertices,
for each vertex v ≠ s

Problem 3-(1). After each iteration, show the D values of all nodes (initial D values are shown above each node in red).
Problem 3-(2). Show the shortest path from S to every other node generated by the algorithm

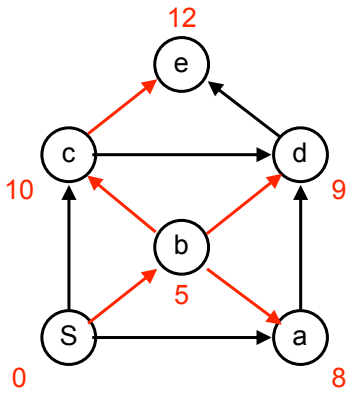
S comes into C,
edges (s, a), (s, b),
and (s, c) are relaxed



a, b, and c comes into C,
edges (a, b), (b,c), (b, d),
and (c, e) are relaxed.



d and e comes into C,
no edge relaxation
needed. Finished.



Problem 3-(2)

S	a	b	c	d	e	iteration
0	∞	∞	∞	∞	∞	
0	16	5	12	∞	∞	1
0	8	5	10	9	∞	2
0	8	5	10	9	∞	3
0	8	5	10	9	∞	4
0	8	5	10	9	12	5

Shortest path
1. S to a: S=>b=>a
2. S to b: S=>b
3. S to c: S=>b=>c
4. S to d: S=>b=>d
5. S to e: S=>b=>c=>e

The correct order of nodes coming into the cloud is: s, b, a, d, c, e

After 1st iteration (S comes into the cloud): S: 0, a: 16, b: 5, c: 12, d: ∞, e: ∞

After 2nd iteration (b comes into the cloud): S: 0, a: 8, b: 5, c: 10, d: 9, e: ∞

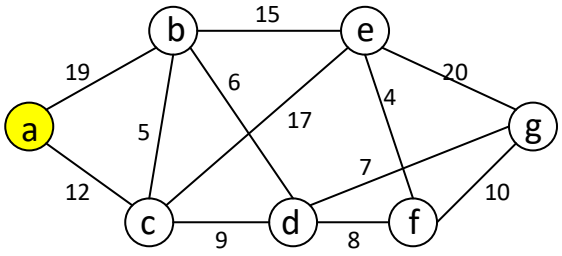
After 3rd iteration (a comes into the cloud): S: 0, a: 8, b: 5, c: 10, d: 9, e: ∞

After 4th iteration (d comes into the cloud): S: 0, a: 8, b: 5, c: 10, d: 9, e: ∞

After 5th iteration (c comes into the cloud): S: 0, a: 8, b: 5, c: 10, d: 9, e: 12

After 6th iteration (e comes into the cloud): S: 0, a: 8, b: 5, c: 10, d: 9, e: 12

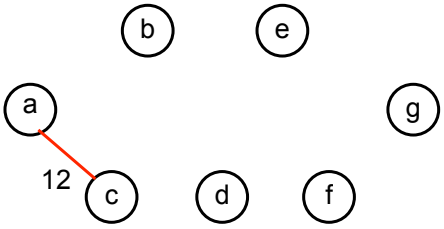
Problem 4 (10 points). Run the Prim-Jarnik algorithm on the following graph beginning at node *a*.



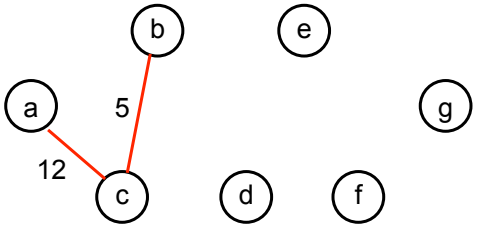
Problem 4-(1). Show the sequence of nodes in the order they are brought into the "cloud."
Problem 4-(2). Show the minimum spanning tree T, generated by the algorithm, as a set of edges.

Result:
4-(1): {a, c, b, d, g, f, e}
4-(2): {(a-c; 12), (c-b, 5), (b-d, 6), (d-g, 7), (d-f, 8), (e-f, 4)}

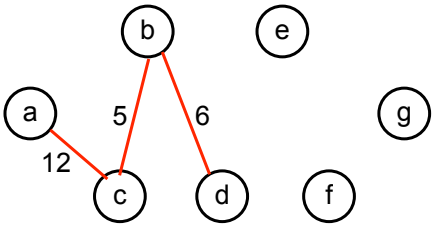
1. (a, c) is a minimum-weight edge



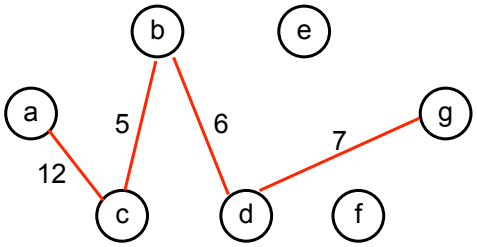
2. (c, b) is a minimum weight edge



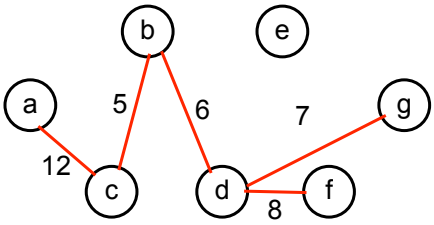
3. (b, d) is a minimum weight edge



4. (d, g) is a minimum weight edge



5. (d, f) is a minimum weight edge



6. (f, e) is a minimum weight edge